Highly Efficient Robot Dynamics Learning by Decomposed Connectionist Feedforward Control Structure

Duško M. Katić and Miomir K. Vukobratović, *Senior Member, IEEE*

Abstract—**A major objective in this paper is the application of connectionist architectures for fast and robust on-line learning of dynamic relations used in robot control at the executive hierarchical level. The proposed connectionist robot controllers as new feature use decomposition of robot dynamics in the space of internal robot coordinates. In this way, this method enables the training of neural networks on the simpler input-output relations with significant reduction of learning time. The proposed controller structure comprises a form of intelligent feedforward control in the frame of decentralized control algorithm with feedback-error or driving torque error learning method. The another important features of these new algorithms are fast and robust convergence properties because the problem of adjusting the weights of internal hidden units is considered as a problem of estimating parameters by recursive least square method and Kalman Filter approach. From simulation examples of robot trajectory tracking it is shown that when a sufficiently trained network is desired, the learning speed of the proposed algorithms is faster than that of the standard back propagation algorithm.**

I. Introduction

THE powerful development of flexible manufacturing systems with high and complex demands for all components of production process, leads toward design of intelligent manipulation robots [1], [2]. We can define these robots as autonomous machines capable of learning, making decisions, exercising fault tolerance, being robust and adaptive to internal and external disturbances, compensating for uncertainties, diagnosing failures, identifying failed components, and recovering from errors. Such robots also, can perform anthropomorphic tasks in an unfamiliar or familiar working environment. There has been a significant effort in making robot more intelligent by integrating advanced sensor systems as vision, tactile sensing, etc. But, one of the major and ultimate steps in robotic research is the formulation and development of intelligent control algorithms which can further improve the performance of robotic systems, using control strategies generated by human intelligent functions as perception, association, reasoning, generalization or learning.

The basic problems for the conventional model-based control of manipulation robots involve the state variables-dependency of the robot dynamic model, the expressive coupling between robot subsystems, coping with structured and unstructured uncertainties and time-dependency of robot parameters. As is well known, none of the conventional robot controllers can not provide desirable solution for these problems, because traditional control laws generally are based on a model that provides incomplete information and only partially known or inaccurately defined parameters. Also, the conventional algorithms are extremely sensitive to the lack of sensor information and to unplanned events and unfamiliar situations in the robot working environment. Robot performance is not able to capture and use past experience and available human expertise. All previously mentioned facts and examples provide a motivation for robotic intelligent control and emphasize the necessity that efficient robotic intelligent control must be based on the learning, generalization and self-organizing capabilities.

The conventional adaptive and non-adaptive control algorithms comprise a robot control problem during execution of single robot trajectories without considering of repetitive motion. Hence, in terms of learning, almost all manipulation robots are memoryless. In this way, the previously acquired experience about dynamic robot model and control algorithms is not applied in robot control synthesis. Using a training process that involves repeating a control task and recording the results accumulated in the entire processii, should thus steadily improve performance. Also, state variables-dependency of robot dynamics may be solved by learning and storing solution, while time-dependency of robot parameters requires an on-line learning approach. If by means of a learning control algorithm, a movement was correctly realized, quite different and faster movement could be controlled using generalization properties of learning algorithm. Hence, one of the primary goals in intelligent control of manipulation robots is the addition of learning and generalization capabilities to the conventional non-adaptive and adaptive control algorithms.

The recent research reports and extensive simulation studies carried out on models containing neural networks have demonstrated an ability to identify and control a sophisticated manipulation robots [3]–[15]. From a systems theoretic point of view, we can say that multilayer neural networks used in robot control in most cases represent static nonlinear mappings as a special part of pattern recognition problem. In this case the patterns to be recognized are the signals of "change" that map in "control action" signals in the aim of desired control goals. The neural network controller should recognize and

Manuscript received October 27, 1991; revised February 1, 1993, February 11, 1993 and July 9, 1993. This work was supported in part by grants from the National Scientific Foundation of Republic Serbia under the project "1004 ROBOTIKA".

The authors are with the Robotics Department, Mihailo Pupin Institute, P.O. Box 15, Volgina 15, 11000 Belgrade, Serbia.

IEEE Log Number 9400637.

isolate signals of "change" in real-time conditions, and using learning by experience and generalization properties, to control efficiently system behavior. Through the training process, the model uncertainties are eliminated, and thus, neural networks serve as a compensation tools in control systems.

In this paper, our purpose is presentation of new robot control learning algorithms with fast and robust learning properties using special connectionist (neural network) architectures. The major concern is the application of neural networks in robot control at executive hierarchical level (motion control problem) for learning inverse dynamic model of robot mechanism in the case where exact robot dynamics is generally unknown.

The several neural network models and learning schemes were applied to learning of robot dynamics, recently. The one of main distinction between these methods is in extent of the knowledge about dynamic models which is used in design procedures. In this context, the term "knowledge" is related for the type and structure of the analytical form of a robot dynamic model. Some methods use in design procedure complete available information about robot model [10]–[12]. The fundamental approach from this group is, for example, that of Kawato [10], [12], who uses a neural structure based on a nonrecurrent single-layer feedforward neural network. This approach has a deterministic nature, but there are several drawbacks related primarly to the inherent complexity of the implementation of a complete model of robot dynamics and poor generalization properties. At the other side of dynamic connectionist approaches are methods that use "black-box" approach in design of neural network algorithms for robot dynamic control [3], [5], [8], [9], [14], [15]. In this case, neural networks can be used as very general computation models. Although a pure neural network approach without knowledge about robot dynamics may be promising, it is important to notice that this approach will not be very practical due high dimensionality of input-output spaces. In this way the training by pure connectionist models would require a neural network of impractical size and unreasonable number of repetition cycles.

Hence, proposed connectionist controller in this paper as new feature uses principle of decomposition of robot dynamics in the space of internal robot coordinates. The results are simplified dynamic mappings and learning feasibility of robotic dynamics for larger systems with a significant reduction of learning time. This method includes a priori knowledge about robot dynamics, which instead of being particular knowledge corresponding to a certain class of models, incorporates general knowledge of robot dynamics.

The other important feature of this new control structure is fast convergence properties, because the problem of adjusting the weights of internal hidden units is considered as a problem of estimating parameters by recursive least square method (RLS) in deterministic case and Kalman filter (KF) approach in stochastic case. In this way, special new algorithms with time-varying learning rate can yield benefits for learning speed and generalization. Also in this paper, adaptive gain of activation functions [16] in learning algorithm is included. It is shown that new learning rules with respect to adaptive gain greatly increases learning speed.

The final result of approach proposed in this paper is a trainable robot controller architecture that uses a neural network model as a form of intelligent feedforward control in the frame of decentralized control algorithm with training by a feedback-error learning method .

II. Robot Control Problem at Executive Hierarchical Level and Connectionist Solution

In contemporary robotic systems, there is a need for more flexible and robust robot controllers in order to take full advantage of the inherent flexibility and versatility of manipulation robots. Difficulties in the solution of control problem arise in various ways.

The one of the most important problems is high nonlinearity of robot dynamics model with expressive couplings between robot subsystems. Based on well-known equations of rigid body mechanics, a dynamic model of manipulation robot in the absence of friction and other disturbances (deterministic model) can be written as

$$P = f(q, \dot{q}, \ddot{q}, \theta) = H(q, \theta)\ddot{q} + h(q, \dot{q}, \theta) \tag{1}$$

or

$$P = f(q, \dot{q}, \ddot{q}, \theta) = H(q, \theta)\ddot{q} + \dot{q}^T C(q, \theta)\dot{q} + g(q, \theta) \tag{2}$$

where $P \in R^n$—is the vector of driving torques or forces; $H(q, \theta) : R^n \times \theta \Rightarrow R^{n \times n}$ is the inertial matrix of the system; $h(q, \dot{q}, \theta) : R^n \times R^n \times \theta \Rightarrow R^n$ is the vector which includes Centrifugal, Coriolis and gravitational effects; $C(q, \theta) : R^n \times \theta \Rightarrow R^n \times R^n \times R^n$ is the matrix which includes Centrifugal and Coriolis effects; $g(q, \theta) : R^n \times \theta \Rightarrow R^n$ is the gravitational vector; $\theta \in R^{nt}$ is the system parameter vector; n—is the number of degree of freedom; nt—is the number of system parameters.

A common and conventional way for robot control represents local PID regulators for each degree of freedom of the robotic mechanism [17]

$$u = u_{fb} = -KP\,\varepsilon - KD\,\dot{\varepsilon} - KI \int \varepsilon dt \tag{3}$$

where $u \epsilon R^n$ is the control input; $u_{fb} \epsilon R^n$ is the feedback control; $KP \epsilon R^{n \times n}$—is the matrix of local position feedback gains; $KD \epsilon R^{n \times n}$—is the matrix of local velocity feedback gains; $KI \epsilon R^{n \times n}$—is the matrix of local integral feedback gains; $\varepsilon = q - q_d$—is the feedback error ($\varepsilon \epsilon R^n$); q and q_d are the real and nominal internal coordinates ($q \epsilon R^n$, $q_d \epsilon R^n$).

However, this control law is not adequate for advanced industrial robots that must demonstrate a capability for high precision and speed in a complex working environment. The influence of couplings between the subsystems is substantial, and we have to include as a solution "dynamic" control [17], which takes the dynamic model of robot mechanism in control synthesis as a form of feedforward control. On the basis of the above, we can apply a *decentralized control algorithm* [17]:

$$u = u_{ff} - KP\,\varepsilon - KD\,\dot{\varepsilon} - KI \int \varepsilon dt \tag{4}$$

where $u_{ff} \epsilon R^n$ is nominal centralized or decentralized feedforward control which is off-line synthesized using the integral

robot model (model of mechanism with the model of robot actuators).

Also, as another solution we can apply *computed torque algorithm* for the calculation of command driving torque [17]:

$$P = \hat{H}(q,\theta)[\ddot{q}_d + KP\,(q - q_d) + KD\,(\dot{q} - \dot{q}_d)] + \hat{h}(q,\dot{q},\theta) \quad (5)$$

where $\hat{H}(q,\theta)$ and $\hat{h}(q,\dot{q},\theta)$ are the estimates of $H(q,\theta)$ and $h(q,\dot{q},\theta)$, respectively.

However, in controller design procedure, we have to cope with structured uncertainties (inaccuracies of model parameters and additive disturbances), unstructured uncertainties (umodelled high frequency dynamics as structural resonant modes, neglected time-delays, actuator dynamics, sampling effects, etc.) and measurement noise. Also, time-varying nature of robot parameters and variability of robot tasks represent additional difficulties for the control system. In this case, the conventional non-adaptive algorithms are not robust enough, because these algorithms compensate only a small number of the uncertainties noted. Hence, a more suitable approach would be the one using adaptive control techniques. The adaptive control techniques in robotics can be applied as a form of the well-known Model-Referenced Adaptive Control (MRAC) or Self-Tuning method [17], with the possibility of adaptation in feedforward or feedback loops. All such methods usually comprise the on-line schemes with recursive least-squares optimization criterion and short-term learning in which the past experience is completely forgotten.

To summarize: Conventional adaptive control techniques robotics are efficient in the case of compensation of structured uncertainties, but in the presence of sensor data overload, heuristic sensor information, limits on real-time applicability and wide interval of unstructured uncertainties, the application of adaptive control is not sufficient for high-quality performance.

Therefore, a solution to the robot control problem will probably require combininig conventional approaches with new learning approaches in order to achieve good performance. For the robot control problem and learning we can identify three main paradigms:

a) Iterative-analytical methods [18], [19];
b) Tabular methods [20];
c) Connectionist methods [3–15].

Iterative-analytical methods are based on successive attempts at following the same trajectory when control input values for each time instant in the trajectory are adjusted iteratively on the basis of the observed trajectory errors at similar time instants during previous attempts. These iterative learning algorithms may be very precise with fast convergence properties. But, on the other hand, drawback of such control techniques is that they are only applicable to the operations which are repetitive. Also, these algorithms have no capability of generalization on quite different movements.

Tabular methods use the associative content—addressable memories, when the robot models are learned by storing experience about command signals and current state coordinates in a memory. Each time a particular set of robot positions, velocities and accelerations is requested, the entire memory has to be searched for the closest experience. In this approach, the problems are great search time due great number of stored experience, the ways to measure similarity, and the methods of efficient generalization.

Connectionist methods with distributed processing provide the implementation tools for complex input/output relations of robot's dynamics and kinematics. One of the main goals of dynamic learning methods is to find solution for the robot inverse dynamic problem. Let us explain the inverse dynamic problem of robot control in a computational framework. There are causal relation between robot driving torque and the resulting robot movement coordinates. Let $P(t)$ denote the time history of driving torque and $q(t)$ denote the time history of the robot internal coordinates during the trajectory. Also we can denote the causal relation between P and q using the functional F, i.e. $F(P(\cdot)) = q(\cdot)$. If we want the robot to tracks desired trajectory q_d, the problem to generate a desired driving torque P_d which realizes q_d, is equivalent to finding an inverse mapping of the functional F. The connectionist approach may, in principle, solve the problem of variable-coupling complexity and state-dependency of robot dynamic model, because neural networks through the process of training can approximate input-output mappings. In this way connectionist structure as part of decentralized feedforward control law can compensate wide range of robot uncertainties. Also, learning by neural networks is based on excellent association and generalization properties. The fast computational capability of neural networks enables real-time applicability of robot control algorithms.

However, there are some problems in applying of the connectionist approach in robot control. First, there is no quarantee in the learning processs for convergence of error to a local minimum. Second, neural networks implement only an approximation of inverse mapping of functional F, so the accuracy and robustness of this approximation may be questionable. Third, there is no systematic way for determinating the optimal topology of connectionist structure (number of layers and neural units, choice of activation functions, convergence parameters, etc.). Also, there are problems in specifying the "good" teacher ("good" basic control algorithm) for supervised learning robot control and specifying the optimal set of input patterns for efficient generalization. One of main problems involves acceleration of the learning rules because connectionist structure may require a long training period to converge. All these problems are open field for connectionist and robotic researchers. Hence, our intention in the following sections is to provide solution to some of the above problems to achieve highly efficient robot learning control.

III. Connectionist Robot Controller with Decomposed Structure

It is important to notice that the application of connectionist solution for robot dynamic learning is not limited only to the task with holonomic constraints (pick-and-place operations and sliding assembly). It is also applicable for the tasks with nonholonomic constraints (mobile robot navigation, grinding

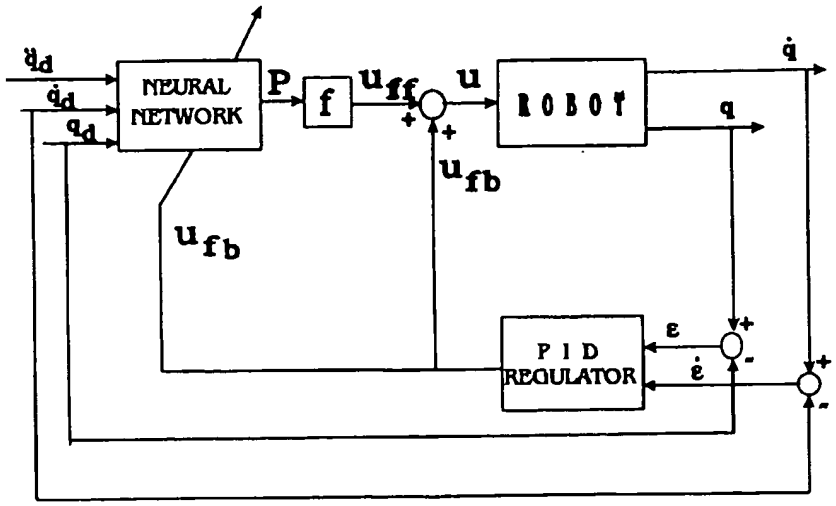

Fig. 1. Decentralized control structure with connectionist feedback error-learning.

and contour following, robot grading, dexterous manipulation or assembly with a robotic hand). There are a few interesting connectionist solutions in this area [21], [22].

However, the primary interest in this paper is the learning of inverse dynamic model of robot mechanism for the tasks with holonomic constraints, where exact robot dynamics is generally unknown. Hence, the proposed neural network models can be regarded as examples of the autonomous driving torque generator (Fig. 1). This connectionist structure is commonly used as part of feedforward controller in decentralized control algorithm. In this case, the feedback controller serves as a robust controller with aim to achieve low errors and perform high-quality learning, because the feedforward controller alone is not sufficient for accurate tracking.

Training and learning by proposed connectionist structure is accomplished exclusively in on-line regime by feedback-error or driving torque error learning method [11] (Fig. 1). This method is exclusively on-line method for robot control, but this control structure provides an internal teacher so that the control scheme works in an unsupervised manner, because we have no external teacher in this case. The adjustment of the network weights during the real-time control by feedback-error learning is more convenient than other learning structures as generalized or specialized learning [5].

The topology of proposed connectionist structure for robot learning control is defined by a four-layer perceptron with *symmetric sigmoid* function as activation functions in both hidden layers. The network has input layer with $3n$ neurons and an output layer with n neurons. This number of neurons in appropriate layers is determined according to number of degree of freedom for standard robot configuration. The activation function for input and output layer is the identity function. The number of neurons in the hidden layers is determined by simulation experiments and experience ($12n$ neurons in the first hidden layer; $6n$ neurons in the second hidden layer). The neural network with proposed topology of fixed nonrecurrent multilayer network generates necessary driving torques in robot joints as a nonlinear mapping of robot desired internal coordinates, velocities and accelerations:

$$P_i = g(w_{jk}^{ab}; q_d, \dot{q}_d, \ddot{q}_d) \quad i = 1, \ldots, n. \tag{6}$$

where $P_i \epsilon R^n$ is joint driving torque generated by neural network; w_{jk}^{ab}—are adaptive weighting factors between neuron j in a-th layer and neuron k in b-th layer; g—nonlinear mapping. According to the integral model of robotic systems, the decentralized control algorithm with learning has the form

$$u_i = f_i(q_d, \dot{q}_d, \ddot{q}_d, P) - KP_{ii}\varepsilon_i - KD_{ii}\dot{\varepsilon}_i - KI_{ii}\int \varepsilon_i dt \quad i = 1, \ldots, n. \tag{7}$$

where f_i is the nonlinear mapping which describes nature of robot actuator model.

Training and learning of proposed connectionist structure can be accomplished using well-known *back propagation algorithm* [23]. In the process of training we can use two type of output error for back propagation algorithm. The first type of error is feedback control signal

$$e_i^{bp} = u_{fb}^i \quad i = 1, \ldots, n. \tag{8}$$

where $e_i^{bp} \epsilon R^n$ is the output error for back propagation algorithm.

But, in fact when we consider integral modelling of robot mechanism with robot actuators model, feedback control signal is not an output error for a neural network. Thus, we must to measure real driving torque or to calculate driving torque error signal

$$e_i^{bp} = P_i^r - P_i = a^i\ddot{\varepsilon}_i + b^i\dot{\varepsilon}_i + c^i u_{fb}^i \quad i = 1, \ldots, n \tag{9}$$

where $P_i^r \epsilon R^n$ is the real driving torque in the robot joints; $a^i \epsilon R^n$, $b^i \epsilon R^n$, $c^i \epsilon R^n$ are parameters of robot integral model

(proposed calculation of driving torque error is valid for commonly used robot DC-actuators).

Although the proposed pure or naive neural network approach without using knowledge about robot dynamics may be promising, it is important to notice that this approach will not be very practical because of high dimensionality of input-output spaces and long learning time. For example, with standard manipulation robots having 6 d.o.f., we have 18 input variables and 6 output variables. Also, the number of trajectory patterns np (input and output variables) may be very large (10–100) if we want to examine the whole working space. Hence, for the robot training it would be necessary to present np^{18} samples, i.e the training by pure connectionist models would require a neural network of impractical size and unreasonable number of repetition cycles. Therefore, we can conclude that the naive connectionist approaches are only applicable for the low-dimension robotic systems.

Bassi, Bekey and Yeung [8], [9], [24] use the principle of *functional decomposition* to simplify the robot dynamics learning. This method includes a priori knowledge about robot dynamics which, instead of being particular knowledge corresponding to a certain type of robot models, incorporates the common knowledge of robot dynamics. In this way, the unknown input-output mapping is decomposed into simpler functions which are easier to learn because of the smaller domains. The process of decomposition has two stages [8]. The first simplification reduces three-vector dynamic function (external accelerations, internal positions and velocities) into the learning of several simpler two-vector basic subfunctions. The feedforward controller using standard back-propagation method can be constructed by combining the particular networks according to their basic subfunctions.

There are some problems in the application of this decomposition. First, the external and internal robot variables are mixed in basic dynamic equations and in this way the separation of variables may be very hard and artificial. Also, the learning domains are still large. Hence, in the second stage of decomposition, full decomposition is applied using a priori knowledge of basic subfunctions. The basic subfunctions for the fixed internal coordinates q can be composed from simpler mappings: constant, linear and quadratic terms. These simple subfunctions may be easily implemented in the feedforward controller using the *context-sensitive approach* [9].

The goal of our approach in synthesis of connectionist robot controller is to leave the control synthesis without a priori information about robot dynamics. Hence, it is very important to use all the available information regarding robot dynamic but only in general and specific form. The general knowledge for that purpose is conveniently incorporated into the structure of network. The way of attaining above goals is a decomposition of robot dynamics on simpler robot dynamic relations in the space of robot internal coordiantes. In this way, instead using a single neural network, training and learning is accomplished by several neural subnetworks which have simpler input-output relations that enable significant reduction of learning time.

Two different methods of robot dynamics decomposition in the space of internal robot coordinates are proposed. We can see that in robot dynamics several terms can be identified which have a distinctive functional dependency. The first method is *"3F-2SF" decomposition* (decomposition of three-vector function (6) into two-vector subfunctions). Exactly, basic robot model (1) can be decomposed into two terms:

first term $H(q,\theta)\ddot{q}$ or $F_1(q,\ddot{q},\theta)$
second term $h(q,\dot{q},\theta)$ or $F_2(q,\dot{q},\theta)$

With this type of decomposition, instead of a multilayer perceptron with $3n$ input values (6), we have two multilayer perceptrons with $2n$ inputs and n outputs for approximation of mapping F_1 and F_2:

$$P_i^{NN1} = F_1(w_{jk}^{NN1ab}, q_d, \ddot{q}_d) \quad i = 1,\ldots,n. \tag{10}$$

$$P_i^{NN2} = F_2(w_{jk}^{NN2ab}, q_d, \dot{q}_d) \quad i = 1,\ldots,n. \tag{11}$$

$$P_i = P_i^{NN1} + P_i^{NN2} \quad i = 1,\ldots,n. \tag{12}$$

where F_1 is a nonlinear mapping for the first perceptron NN1; F_2 is a nonlinear mapping for the second perceptron NN2; P_i^{NN1} and P_i^{NN2} are parts of robot dynamic model generated by perceptrons $NN1$ and $NN2$; w_{jk}^{NN1ab} and w_{jk}^{NN2ab} are weighting factors for perceptrons $NN1$ and $NN2$; P_i is driving torque at the output of connectionist structure.

The topology of perceptrons NN1 and NN2 are determined using similar activation functions and principles as in the previous case (input layer-$2n$ neural units; first hidden layer-$8n$ neural units; second hidden layer-$4n$ neural units; output layer-n neural units). Training of both perceptrons is accomplished synchronously by feedback-error learning method (Fig. 2). The feedback error signal or driving-torque error signal is transferred as output backpropagation error to the both perceptron outputs.

The second method of decomposition includes deeper decomposition. It is *"3F-1SF" decomposition* (decomposition of three-vector function into three one-vector subfunctions). The robot model (2) can be decomposed into three terms:

first term $H(q,\theta)\ddot{q}$ or $F_1(q,\ddot{q},\theta)$
second term $g(q,\theta)$ or $F_2(q,\theta)$
third term $\dot{q}^T C(q,\theta)\dot{q}$ or $F_l(q,\dot{q},\theta) \quad l = 3,\ldots,n+2.$

We can see that each of these terms represents multiplications of an internal position-vector subfunction by robot internal variables or internal position subfunction itself, as is case for the third term. Our intention is to use multilayer perceptrons for learning the nonlinear position-vector subfunctions $H(q,\theta)$, $C(q,\theta)$ and $g(q,\theta)$. The total number of multilayer perceptrons in this case is $n+2$ because position subfunctions $C(q,\theta)$ has dimension $n \times n \times n$ (to be precise, we have n subfunctions $C(q,\theta)$ with dimension $n \times n$). The multilayer perceptrons and the driving torque at the output of the connectionist structure are defined according to the following equations:

learning of $H(q,\theta)$

$$P_{il}^{NN1} = F_1(w_{jk}^{NN1ab}, q_d) \quad i = 1,\ldots,n. \quad l = 1,\ldots,n. \tag{13}$$

learning of $g(q,\theta)$

$$P_i^{NN2} = F_2(w_{jk}^{NN2ab}, q_d) \quad i = 1,\ldots,n. \tag{14}$$

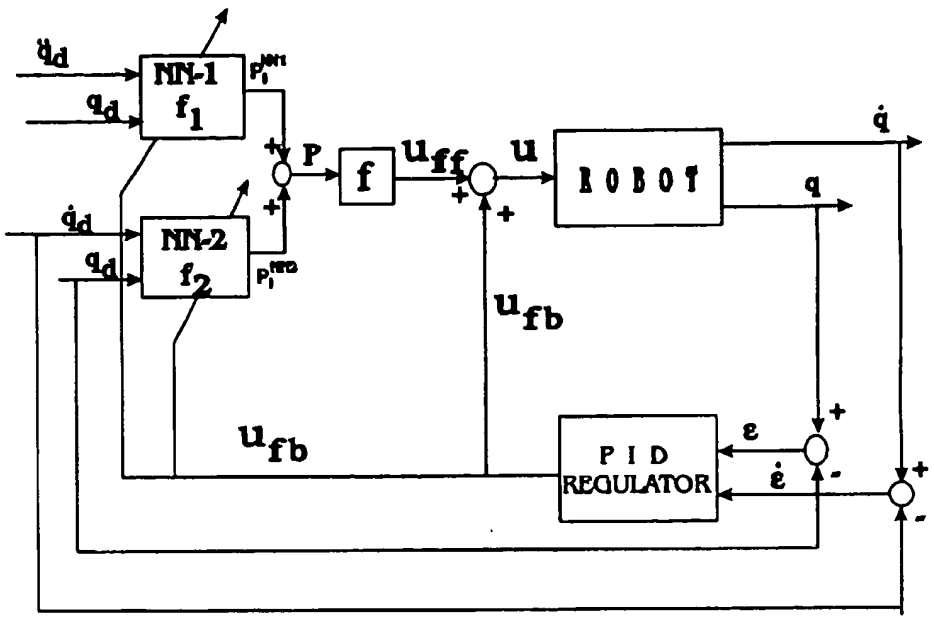

Fig. 2. "3F-2SF" Decomposition connectionist structure with feedback error learning.

learning of $C^l(q,\theta)$

$$P_{im}^{NNl} = F_l(w_{jk}^{NNlab}, q_d) \quad i = 1,\ldots,n. \quad m = 1,\ldots,n. \quad l = 3,\ldots,n+2. \tag{15}$$

total output of connectionist feedforward structure

$$P_i = \sum_{l=1}^{n} P_{il}^{NN1}\tilde{q}_l + \dot{q}^T P^{NNi+2}\dot{q} + P_i^{NN2} \tag{16}$$

where $f_i(i = 1,\ldots,n+2)$ is a nonlinear mapping for perceptron NNi; P_i^{NNl} and P_{im}^{NNl} are the outputs of the perceptron NNl $(i = 1,\ldots,n;\ m = 1,\ldots,n;\ l = 1,\ldots,n+2;)$; w_{jk}^{NNlab} are the weighting factors for perceptron NNl.

In this case, number of neural units in network layers is defined according to specific features of robot dynamic models. For the first perceptron, which has a role to approximate $H(q,\theta)$, we have the next topology: input layer-n neural units; first hidden layer-$2n(n+1)$ neural units; second hidden layer—$n(n+1)$ neural units; output layer-$n(n+1)/2$ neural units. Number of neural units in the output layer is defined according to number of elements of the inertia matrix $H_{ii}(q,\theta)$ using symmetry properties of matrix $H(q,\theta)$. The topology of perceptron $NN2$ which we are using for approximation of vector $g(q,\theta)$ is: input layer-n neural units; first hidden layer-$4n$ neural units; second hidden layer-n neural units. Finally, the topology of n perceptrons for learning of subfunctions $C^l(q,\theta)$ is: input layer-n neural units; first hidden layer-$2n(n+1)$ neural units; second hidden layer—$n(n+1)$ neural units; output layer-the number of neural units in this layer is different between these n perceptrons due properties of symmetry and antisymmetry. The whole number of outputs for all these n perceptrons is $n(n-1)/3$.

Training of both perceptrons is accomplished synchronously using simlar principles as in previous cases by feedback-error learning method (Fig. 3).

The decomposition steps of robot dynamics model with proposed neural network structure (number of neural units in network layers) are illustrated on Fig. 4. The bias neural unit is included in all layers except for the output layer. In this particular case, number of the neurons in the output layers for *"3F-1SF"* decompositions is determined without using properties of symmetry and antisymmetry.

It is important to notice that in both case of decomposition, outputs of connectionist structure defined by (12) and (16) are part of decentralized control algorithm according to (7).

The main result of these decompositions is a significant reduction in the size of the trajectory input patterns. For example, we can perform learning analysis for standard robot configurations with 3-d.o.f.(degree of freedom) and 6 – d.o.f. by knowing the number of different patterns necessary to train the proposed neural network structures. The number of patterns is determined by number of samples np per each variable. In Table I are shown one example of learning analysis with the number of samples for 3 – d.o.f. and 6-d.o.f. manipulators using $np = 10$ samples per dimension.

It is evident from Table I that the reduction in the number of patterns required for learning is $1 \cdot 10^6$ for $n = 3$ and $1 \cdot 10^{12}$ for $n = 6$. On this way, breaking of a large space into smaller ones allows more practical learning the complex dynamics of high dimensional robotic systems. At the other side, instead of learning on single perceptron, we have simultaneous learning on several perceptrons. Hence, the real advantage of decomposed structure is a strong connection with

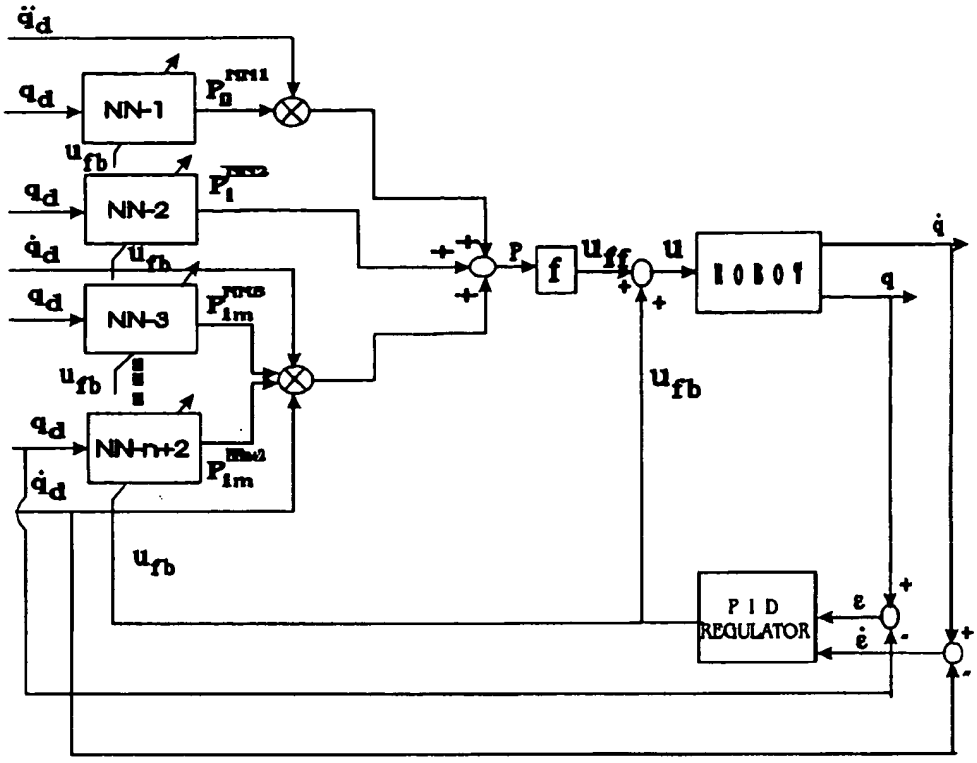

Fig. 3. "3F-1SF" Decomposition connectionist structure with feedback error learning.

TABLE I
EXAMPLE OF NUMBER OF PATTERNS NEEDED FOR 3-d.o.f AND 6-d.o.f ROBOTS

Method	Number of samples	$n = 3$ $np = 10$	$n = 6$ $np = 10$
pure connect. method	np^{3n}	10^9	10^{18}
"3F-2SF" decomp. connect.	$2np^{2n}$	$2 \cdot 10^6$	$2 \cdot 10^{12}$
"3F-1SF decomp. connect.	np^n	10^3	10^6

concurrent distributed processing nature of neural networks. With this goal, it is important to use on contemporary robot controllers a highly efficient parallel processing algorithms with real concurrent neural network hardware architectures [25]–[28].

We can conclude that when we use some aspect of robot dynamics information, principle of decomposition in the space of internal robot coordinates is a proper approach to real-time learning.

IV. New Connectionist Feedforward Control Algorithms With Fast Learning Properties

The back propagation algorithm caused a significant breakthrough in the control application of multilayer preceptrons. The back propagation learning algorithm for layered neural nets performs a gradient search in the weight space, minimizing cost function. One of the major drawbacks of this method is its slow convergence. Starting from a random initial state, the path to the global minimum is often strewn with local minimum, causing oscillations around ravines in the weight space. Sometimes the algorithm gets stuck in a local minimum and the training has to be repeated, starting from another initial state.

There is a huge literature in the neural network field [16], [29]–[32] on how to speed up backpropagation. The one of common way to attaining this goal is by means of various methods from the field of numerical analysis. For example, it is very promising to use a modified version of error back propagation algorithm based on conjugate gradient descent method, because this method has quadratic convergence properties.

In this paper, our intention is that instead acceleration of standard back propagation algorithm using standard methods from numerical analysis, consider the problem of adjusting the weights of internal hidden units as a problem of parameter estimation for linear deterministic systems by Recursive Least Square (RLS) method or as a filtering problem for linear stochastic systems by Kalman Filter (KF) approach [33]. Using these methods with time-varying learning rate yield to benefits for learning speed and generalization than are available with the standard back propagation algorithm.

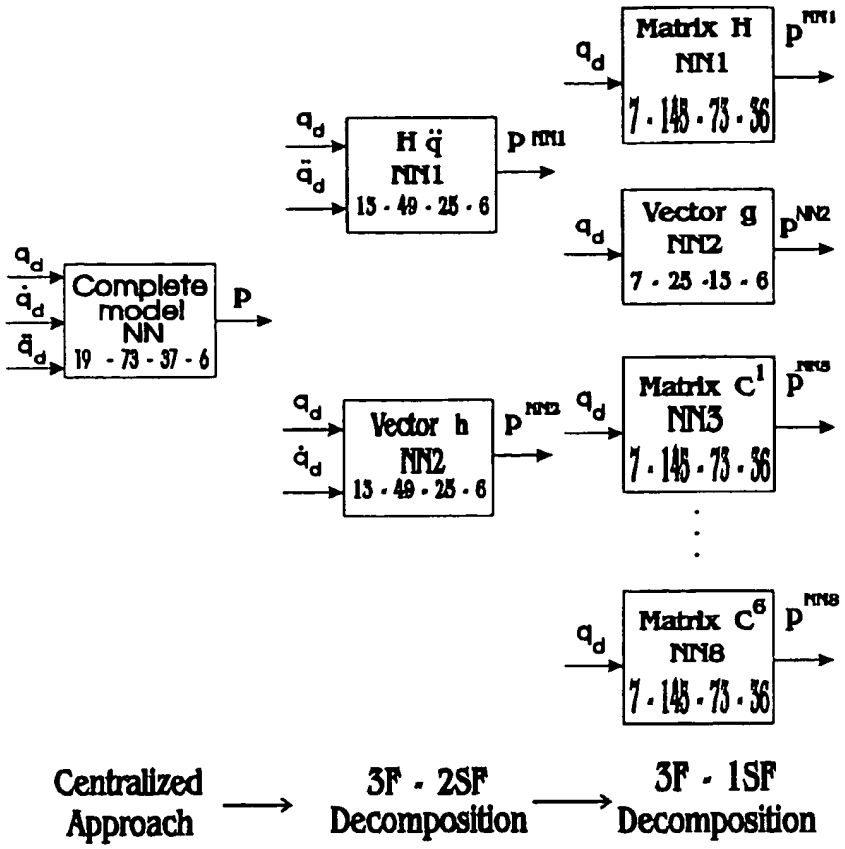

Fig. 4. Decomposition steps of robot dynamics by connectionist structures.

The another important feature for acceleration of learning algorithm is inclusion of adaptive gain in activation function for hidden layers [16]. The adaptive gain of a node in a connectionist network is a multiplicative constant that amplifies or attenuates the net input to the node. It is shown that using of gradient descent with respect to gain greatly increases learning speed by amplifying those directions in weight space that are successfully chosen by gradient descent on weights [16].

The proposed new algorithms are based on previously defined four-layer decomposed connectionist structures ("*3F-2SF*" and "*3F-1SF*") with appropriately defined parameteres of network.

The general forward network relations in process of training are described according to the next expressions:

Forward Relations RLS & KF method

$$s^2(k) = W^{12}(k)^T i^1(k) \quad i_0^1(k) = 1 \tag{17}$$

$$o_a^2(k) = 1/(1 + \exp(-g_a^2(k)s_a^2(k))) - 0.5 \quad a = 1, \ldots, L_1 \quad o_0^2(k) = 1 \tag{18}$$

$$s^3(k) = W^{23}(k)^T o^2(k) \tag{19}$$

$$o_b^3(k) = 1/(1 + \exp(-g_b^3(k)s_b^3(k))) - 0.5 \quad b = 1, \ldots, L_2 \quad o_0^3(k) = 1 \tag{20}$$

$$s^4(k) = W^{34}(k)^T o^3(k) \tag{21}$$

$$y_c(k) = s_c^4(k) = P_c^{NNp}(k) \quad \text{or} \quad P_{cl}^{NNp}(k) \quad c = 1, \ldots, n \quad l = 1, \ldots, n \quad p = 3, \ldots, n+2 \tag{22}$$

where $s^2(k)$, $s^3(k)$, $s^4(k)$—are the output vectors for linear parts of layers; $o^2(k)$, $o^3(k)$—are the output vectors of the hidden layers; $g_a^2(k)$, $g_b^3(k)$—are the adaptive gains of activation functions for hidden layers; $W^{12} = [w_{m\times L1}^{12}]$, $W^{23} = [w_{L1+1\times L2}^{23}]$, $W^{34} = [w_{L2+1\times n}^{34}]$ are the weighting factors of the layers; $i^1(k)$ is the input of network (robot internal positions, velocities and accelerations—$m = 3n + 1$); $y(k)$ is output of network.

As first approach for estimating the weights at the hidden layers, RLS method for parameter estimation of linear systems is applied. The aim of estimation is to define optimal values for matrices W^{12}, W^{23}, W^{34} using models of linear systems according to equations (17), (19), and (21). In application of this method, problem of specification of values for desired outputs of linear subsystems is arose. This problem is solved by inverse mapping of values for desired outputs of nonlinear

sybsystems (18) and (20) [34]. As second important feature, updated values of weighting factors in current iteration are back propagated to the input layer. Beside the on-line learning, the proposed algorithm is valid for off-line learning of robot dynamics as well as for any other mapping problem.

The basic equations which describe new learning rules based on RLS method are given according to the next formulas:

Learning Rules RLS method

$$s_c^{4d}(k) = y_c^d(k) = P_c^r(k) \quad c = 1, \ldots, n \tag{23}$$

$$\delta_c^4(k) = y_c^d(k) - y_c(k) = P_c^r(k) - \sum_j w_{jc}^{34}(k) o_j^3(k)$$
$$c = 1, \ldots, n \quad j = 1, \ldots, L_2 + 1 \tag{24}$$

$$P^3(k) = \frac{1}{\lambda^3}\left\{P^3(k-1) - \frac{P^3(k-1)o^3(k)o^3(k)^T P^3(k-1)}{\lambda^3 + o^3(k)^T P^3(k-1) o^3(k)}\right\} \tag{25}$$

$$W^{34}(k) = W^{34}(k-1) + P^3(k)o^3(k)[s^{4d}(k) - W^{34}(k-1)^T o^3(k)]^T \tag{26}$$

$$\delta_c^4(k) = s_c^{4d}(k) - \sum_j w_{jc}^{34}(k) o_j^3(k)$$
$$c = 1, \ldots, n \quad j = 1, \ldots, L_2 + 1 \tag{27}$$

$$o^{3d}(k) = o^3(k) + W^{34}(k)\delta^4(k) \tag{28}$$

$$o_{\max}^{3d}(k) = \max \mid o_b^{3d}(k) \mid \quad b = 1, \ldots, L_2 \tag{29}$$

$$s_b^{3d}(k) = \ln\left[\frac{0.5 + \left(\frac{0.49}{o_{\max}^{3d}}\right) o_b^{3d}(k)}{0.5 - \left(\frac{0.49}{o_{\max}^{3d}}\right) o_b^{3d}(k)}\right] / g_b^3(k) \quad b = 1, \ldots, L_2 \tag{30}$$

$$P^2(k) = \frac{1}{\lambda^2}\left\{P^2(k-1) - \frac{P^2(k-1)o^2(k)o^2(k)^T P^2(k-1)}{\lambda^2 + o^2(k)^T P^2(k-1) o^2(k)}\right\} \tag{31}$$

$$W^{23}(k) = W^{23}(k-1) + P^2(k)o^2(k)[s^{3d}(k) - W^{23}(k-1)^T o^2(k)]^T \tag{32}$$

$$\delta_b^3(k) = s_b^{3d}(k) - \sum_j w_{jb}^{23}(k) o_j^2(k)$$
$$b = 1, \ldots, L_2 \quad j = 1, \ldots, L_1 + 1 \tag{33}$$

$$o^{2d}(k) = o^2(k) + W^{23}(k)\delta^3(k) \tag{34}$$

$$o_{\max}^{2d}(k) = \max \mid o_a^{2d}(k) \mid \quad a = 1, \ldots, L_1 \tag{35}$$

$$s_a^{2d}(k) = \ln\left[\frac{0.5 + \left(\frac{0.49}{o_{\max}^{2d}}\right) o_a^{2d}(k)}{0.5 - \left(\frac{0.49}{o_{\max}^{2d}}\right) o_a^{2d}(k)}\right] / g_a^2(k) \quad a = 1, \ldots, L_1 \tag{36}$$

$$P^1(k) = \frac{1}{\lambda^1}\left\{P^1(k-1) - \frac{P^1(k-1)i^1(k)i^1(k)^T P^1(k-1)}{\lambda^1 + i^1(k)^T P^1(k-1) i^1(k)}\right\} \tag{37}$$

$$W^{12}(k) = W^{12}(k-1) + P^1(k)i^1(k)[s^{2d}(k) - W^{12}(k-1)^T i^1(k)]^T \tag{38}$$

where λ^1, λ^2, λ^3—are the appropriate forgetting factors.

Initial conditions for weighting factors are generated by normal distribution with different random numbers:

$$W^{12}(0) = N^{12}(0,1); \quad W^{23}(0) = N^{23}(0,1); \quad W^{34}(0) = N^{34}(0,1); \tag{39}$$

Also, initial conditions for covariance matrices P are given in the following form:

$$P^1(0) = c^1 I_{m \times m}; \quad P^2(0) = c^2 I_{L_1+1 \times L_1+1}; \quad P^3(0) = c^3 I_{L_2+1 \times L_2+1}; \tag{40}$$

where c^1, c^2, c^3 are positive numbers.

The values of adaptive gains in activation functions are updated according to gradient descent on error that can be computed using standard back propagation algorithm:

Learning Rules Adaptive Gains

$$\Delta_c^4(k) = y_c^d(k) - y_c(k) = P_c^r(k) - P_c(k) = e_c^{bp}(k)$$
$$c = 1, \ldots, n \tag{41}$$

$$op_b^3(k) = \tilde{o}_b^3(k)(1 - \tilde{o}_b^3(k)) \quad b = 1, \ldots, L_2 \tag{42}$$

$$\Delta_b^3(k) = \left(\sum_c \Delta_c^4(k) w_{bc}^{34}(k)\right) g_b^3(k) op_b^3(k)$$
$$b = 1, \ldots, L_2 \tag{43}$$

$$g_b^3(k) = g_b^3(k-1) + \zeta_g^3\left(\sum_c \Delta_c^4(k) w_{cb}^{34}(k)\right) op_b^3(k) s_b^3(k)$$
$$b = 1 \ldots, L_2 \tag{44}$$

$$op_a^2(k) = \tilde{o}_a^2(k)(1 - \tilde{o}_a^2(k)) \quad a = 1, \ldots, L_1 \tag{45}$$

$$g_a^2(k) = g_a^2(k-1) + \zeta_g^2\left(\sum_b \Delta_b^3(k) w_{ab}^{23}(k)\right) op_a^2(k) s_a^2(k)$$
$$a = 1 \ldots, L_1 \tag{46}$$

where ζ_η^2, ζ_g^3 are learning rates for adaptive gains; $\tilde{o}_b^3(k) = o_b^3(k) + 0.5$; $\tilde{o}_a^2(k) = o_a^2(k) + 0.5$.

Using adaptive gains and time-varying learning rate in RLS method enables better accommodation of learning process to the robot dynamics.

The KF approach for linear systems [33] is a second approach applied to estimating weighting factors in the network layers. In references [35], [36] similar strategy but for nonlinear stochastic problems using Extended Kalman Filter approach was considered with application for the classical XOR problem. In our, robotic case, the KF approach is related to linear state estimation ("filtering") problems where the following observation equations are observed :

$$w_{ij}^{ab}(k) = w_{ij}^{ab}(k-1) + \xi(k) \tag{47}$$

$$s_j(k) = \sum_i w_{ij}^{ab}(k-1) o_i(k) + \eta(k) \tag{48}$$

or

$$s_j(k) = \sum_i w_{ij}^{ab}(k-1) i_i(k) + \eta(k) \tag{49}$$

where $\{\xi(k), \eta(k)\}$ are mutually independent, zero-mean white gaussian noises with variances R_1 and R_2. It is important to

notice that these noises can be considered as pseudo-noises for tuning the gain in a Kalman filter.

The learning algorithm via the KF approach by using a new specification for the desired states and errors in hidden layers is defined according to the following expressions:

Learning Rules KF method

$$s_c^{4d}(k) = y_c^d(k) = P_c^r(k) \quad c = 1, \ldots, n \tag{50}$$

$$\delta_c^4(k) = y_c^d(k) - y_c(k) = P_c^r(k) - \sum_j w_{jc}^{34}(k) o_j^3(k)$$

$$c = 1, \ldots, n \quad j = 1, \ldots, L_2 + 1 \tag{51}$$

$$K^3(k) = P^3(k-1) o^3(k) [R_2^3 + o^3(k)^T P^3(k-1) o^3(k)]^{-1} \tag{52}$$

$$P^3(k) = [I - K^3(k) o^3(k)^T] P^3(k-1) + R_1^3 I \tag{53}$$

$$W^{34}(k) = W^{34}(k-1) + K^3(k) [s^{4d}(k) - W^{34}(k-1)^T o^3(k)]^T \tag{54}$$

$$\delta_c^4(k) = s_c^{4d}(k) - \sum_j w_{jc}^{34}(k) o_j^3(k)$$

$$c = 1, \ldots, n \quad j = 1, \ldots, L_2 + 1 \tag{55}$$

$$o^{3d}(k) = o^3(k) + W^{34}(k) \delta^4(k) \tag{56}$$

$$o_{\max}^{3d}(k) = \max \mid o_b^{3d}(k) \mid \quad b = 1, \ldots, L_2 \tag{57}$$

$$s_b^{3d}(k) = \ln \left[\frac{0.5 + \left(\frac{0.49}{o_{\max}^{3d}}\right) o_b^{3d}(k)}{0.5 - \left(\frac{0.49}{o_{\max}^{3d}}\right) o_b^{3d}(k)} \right] / g_b^3(k) \quad b = 1, \ldots, L_2 \tag{58}$$

$$K^2(k) = P^2(k-1) o^2(k) [R_2^2 + o^2(k)^T P^2(k-1) o^2(k)]^{-1} \tag{59}$$

$$P^2(k) = [I - K^2(k) o^2(k)^T] P^2(k-1) + R_1^2 I \tag{60}$$

$$W^{23}(k) = W^{23}(k-1) + K^2(k) [s^{3d}(k) - W^{23}(k-1)^T o^2(k)]^T \tag{61}$$

$$\delta_b^3(k) = s_b^{3d}(k) - \sum_j w_{jb}^{23}(k) o_j^2(k)$$

$$b = 1, \ldots, L_2 j = 1, \ldots, L_1 + 1 \tag{62}$$

$$o^{2d}(k) = o^2(k) + W^{23}(k) \delta^3(k) \tag{63}$$

$$o_{\max}^{2d}(k) = \max \mid o_a^{2d}(k) \mid \quad a = 1, \ldots, L_1 \tag{64}$$

$$s_a^{2d}(k) = \ln \left[\frac{0.5 + \left(\frac{0.49}{o_{\max}^{2d}}\right) o_a^{2d}(k)}{0.5 - \left(\frac{0.49}{o_{\max}^{2d}}\right) o_a^{2d}(k)} \right] / g_a^2(k) \quad a = 1, \ldots, L_1 \tag{65}$$

$$K^1(k) = P^1(k-1) i^1(k) [R_2^1 + i^1(k)^T P^1(k-1) i^1(k)]^{-1} \tag{66}$$

$$P^1(k) = [I - K^1(k) i^1(k)^T] P^1(k-1) + R_1^1 I \tag{67}$$

$$W^{12}(k) = W^{12}(k-1) + K^1(k) [s^{2d}(k) - W^{12}(k-1)^T i^1(k)]^T \tag{68}$$

Initial conditions for weighting factors are generated by normal distribution with different random numbers or using previously "learned values" of weighting factors in off-line learning:

$$W^{12}(0) = N^{12}(0,1); \quad W^{23}(0) = N^{12}(0,1); \quad W^{34}(0) = N^{34}(0,1); \tag{69}$$

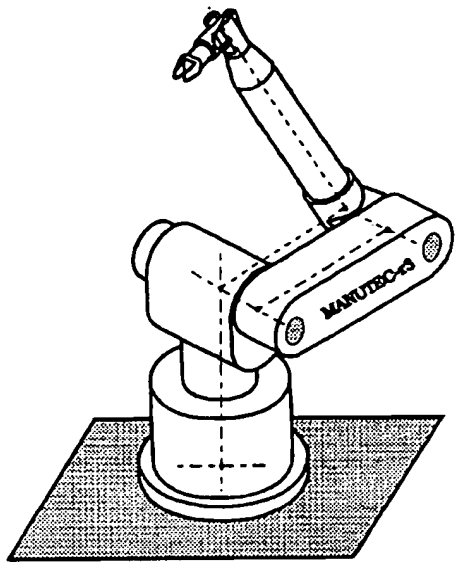

Fig. 5. Industrial robot MANUTEC r3.

Also, initial conditions for covariance matrices P and variances R_1 and R_2 are given in following form :

$$P^1(0) = c^1 I_{m \times m}; \quad P^2(0) = c^2 I_{L_1+1 \times L_1+1}; \quad P^3(0) = c^3 I_{L_2+1 \times L_2+1}; \tag{70}$$

$$R_1^i = c_1^i; \quad R_2^i = c_2^i; \quad i = 1, 2, 3 \tag{71}$$

where c^1, c^2, c^3, c_1^i, c_2^i, c_3^i,$(i = 1, 2, 3)$ are positive numbers.

The values of adaptive gains in activation functions are updated on same way according to previously defined equations (41)–(46). Also, we can see that KF algorithm (50)–(68) is equivalent to RLS algorithm (23)–(38) for $R_1^i = 0.$ and $R_2^i = 1.$

V. Simulation Examples

In this section, simulation trials were run to verify the proposed connectionist algorithms in compensating the system uncertainities. The manipulation robot used for the simulation was a 6 d.o.f. industrial robot MANUTEC r3 [37] (Fig. 5). The parameters of the robot mechanism and actuators are listed in Tables II and III.

In the learning phase, robot training is accomplished using movement from point A with internal coordinates $\{qc(0.; -1.57; 0.; 0.; 0.; 0.)\}$ to point B $\{q\epsilon(0.7; -0.785; 0.5; 0.3; 0.3; 0.3)\}$. Time duration of movement is $t = 1s$ with a trapezoidal velocity profile. The PD feedback control was choosen with next values for local feedback gains: $KP\epsilon(62.04; 63.19; 25.74; 32.22; 27.89; 30.31)$; $KDc(2.88; 2.99; 1.5; 1.24; 1.21; 1.05)$. We know that exact measuring of the link inertia and position of mass centre are very difficult. Hence in simulation experiments, the model uncertainties are defined by parametric disturbances

TABLE II
PARAMETERS OF MANUTEC R3 MANIPULATION MECHANISM

Mechanism link no.	1	2	3	4	5	6
Mass [kg]	50.	56.5	26.4	28.7	5.2	0.01
Length [m]	0.67	0.5	0.73	0.01	0.01	0.01
Moment of inertia $J_x[kgm^2]$	0.	2.58	0.279	1.67	1.25	0.
Moment of inertia $J_y[kgm^2]$	0.	2.73	0.413	1.67	1.53	0.
Moment of inertia $J_z[kgm^2]$	1.16	0.64	0.245	0.81	0.81	0.001

TABLE III
PARAMETERS OF D.C. ACTUATORS FOR ROBOT MANUTEC R3

Actuator	1	2	3	4	5	6
Mechanical constant [Nm/A]	126.	252.	72.	24.8	21.4	8.6
Electricul constant [Vs/rad]	2.5	5.	1.428	2.2	1.74	2.027
M.inertia of rotor $[kgm^2]$	0.0013	0.0013	0.0013	0.00016	0.00018	0.00004
Viscous coefficient [Nms/rad]	1132.979	1091.	411.6	108.	112.6	3.6
Rotor resistance [Ω]	1.	1.	1.	1.	1.	1.
Voltage saturation [v]	7.5	7.5	7.5	7.5	7.5	7.5
Reduction ratio	105.	210.	60.	99.	79.2	99.
Reducer efficiency	1.	1.	1.	1.	1.	1.
Rotor inductivity [H]	0.001	0.001	0.001	0.001	0.001	0.001

with approximately 20% variation from nominal values for link mass and moment of inertia).

In simulation experiments we were applied pure connectionist and decomposed "3F-2SF" network structure . The pure connectionist structure has next topology : input layer-19 neural units with identity activation function; first hidden layer-73 neural units with symetric sigmoid activation function; second hidden layer-37 neural units with symetric sigmoid activation function; output layer-13 neural units with identity activation function. The decomposed structure has a next topology for each of multilayer perceptron: input layer-6 neural units with identity function; first hidden layer-49 neural units with symmetric sigmoid activation function; second hidden layer-23 neural units with symmetric sigmoid activation function; output layer-6 neural units.

It is important to notice that if we use directly in on-line feedback learning randomly initialized set of values for weighting factors, learning may be in some cases very slowly and impractical. Hence, our intention in this case is to use off-line learning with available a priori known informations about system model without model uncertainities. In process of off-line learning, set of weighting factors are generated which represents closed neighborhood of real optimal value for weighting factors. This set represents starting set for learning of real weighting factors in on-line feedback learning. In simulation off-line experiments, we use all previously defined connectionist structures for generating weighting factors that

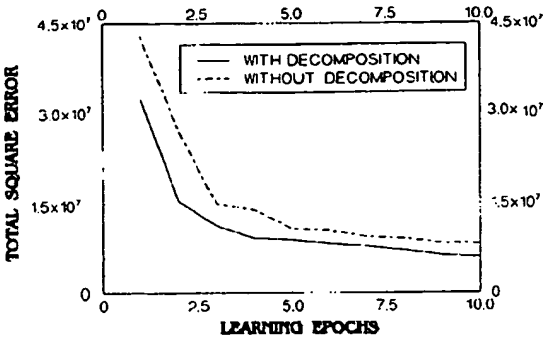

Fig. 6. Convergence results for BP algorithm with and without decomposition.

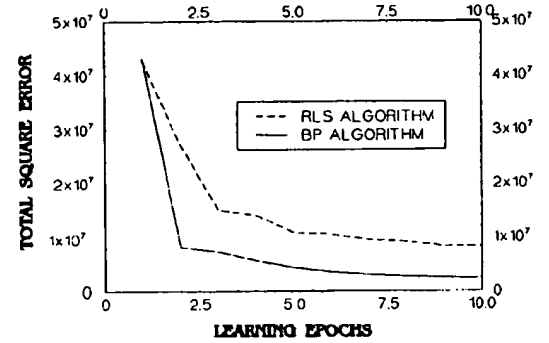

Fig. 7. Total square error during training epochs for back propagation algorithm and RLS method without adaptive gains.

are randomly initialized by Gaussian numbers with zero mean and variance $\sigma = 0.5$. This idea is similar to learning in reference [5] where combined learning methods (generalized and specialized learning method) are used.

The learning of robot dynamics is accomplished through a trial-and-error approach, when in successive epochs of training we present the same trajectory patterns.

In simulation experiment with backpropagation algorithm, the learning rate for all layers is $\eta = 0.01$ while the momentum factor is $\alpha = 0.9$. In the case of RLS connectionist algorithm, covariance matrices have a next initial values:

$$P^1(0) = 1. \quad P^2(0) = 1. \quad P^3(0) = 1. \tag{72}$$

while learning rates for adaptive gains have next values

$$\zeta_g^2 = 0.01 \quad \zeta_g^3 = 0.01 \tag{73}$$

In the case of KF approach, values of variance R_1^i and R_2^i are defined according to next values $R_1^i = 0.01$ and $R_2^i = 1.$ $(i = 1, 2, 3)$, while covariance matrices have a next initial values:

$$P^1(0) = 1. \quad P^2(0) = 1. \quad P^3(0) = 1. \tag{74}$$

When comparing the pure connectionist structure and the *"3F-1SF"* decomposed connectionist structure, simulations ex-

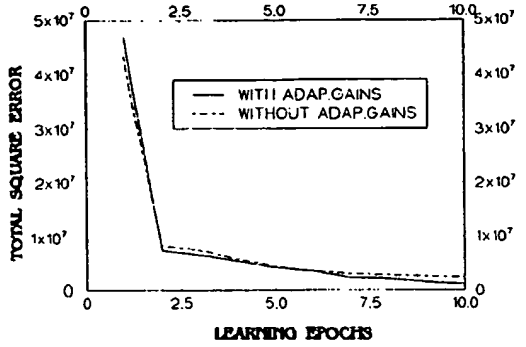

Fig. 8. Total square error during training epochs for RLS algorithm with and without adaptive gains.

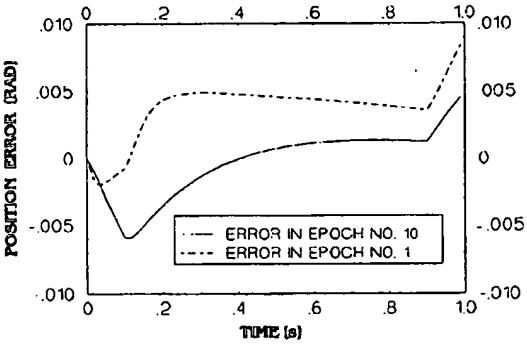

Fig. 9. Position error for first d.o.f.-BP algorithm.

periments with standard back propagation (BP) algorithm under the same conditions are performed. Fig. 6 shows the convergence results (i.e., total epoch square error during time with sampling period $t = 0.1$ s) for both connectionist structure. The result shows better learning of dynamic robot model and significant reduction of learning time for decomposed connectionist structure. The convergence results of back propagation algorithm and RLS algorithm (Fig. 7) in the case of on-line learning but without adaptive gains are compared. Also, on the Fig. 8, total epoch square error in the case of RLS method with and without adaptive gains is presented. We can conclude from results, that especially with new RLS connectionist method and adaptive gains we can achieve the fast learning of robot dynamics.

At Figs. 9 and 10, position error is shown for the first d.o.f. in the case of trajectory tracking with BP algorithm and RLS method (both with adaptive gains). The figure shows the first and 10th training trial. At Figs. 11 and 12, comparison of BP method, RLS method and KF method in on-line learning is given. In all cases, position error fir d.o.f. for 10-th epoch of training is shown.

The figures show that with repetitive trials tracking errors using RLS method are considerably decreased and on this way highly efficient robot dynamic learning is accomplished. Also, with proposed KF method, good results are achieved

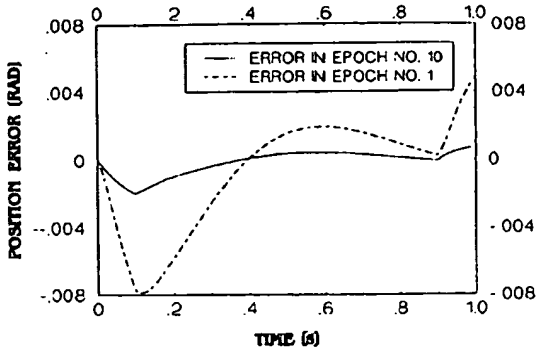

Fig. 10. Position error for first d.o.f.-RLS algorithm.

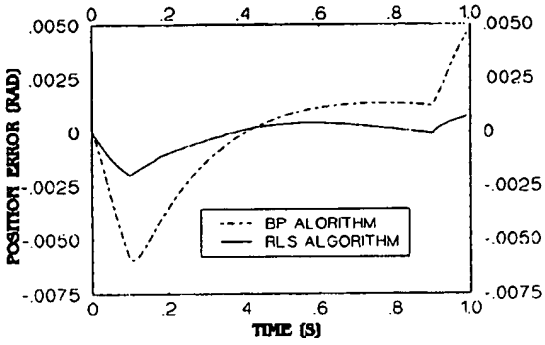

Fig. 11. Comparison of learning algorithms (BP-RLS) for first d.o.f.

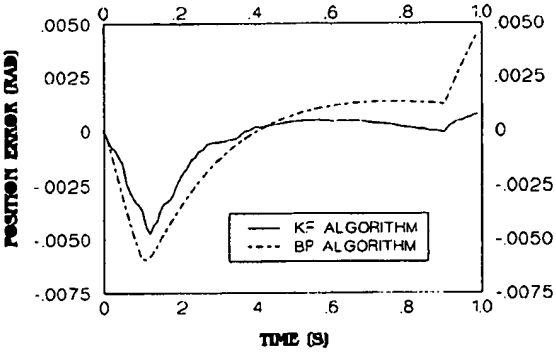

Fig. 12. Comparison of learning algorithms (BP-KF) for first d.o.f.

but it is important to notice that performance of this method extremely depends from initial values of variances R_1^i and R_2^i. With aim to verify generalization properties of proposed algorithms, some simulation experiments are performed. In the generalization phase robot moves along a different trajectory from that used in the learning phase, exactly from point A $q\epsilon(0.36; 0.95; 0.75; 0.1; 0.5; 0.)$ to point $Bq\epsilon(1.25; 0.42; 0.22; 0.; 0.95; 0.1)$ with triangular velocity profile.

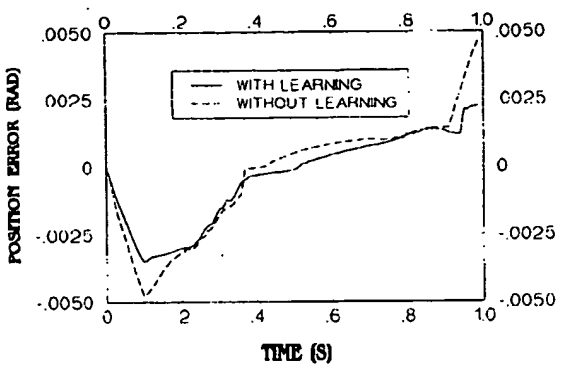

Fig. 13. Position error for the first d.o.f. in generalization phase.

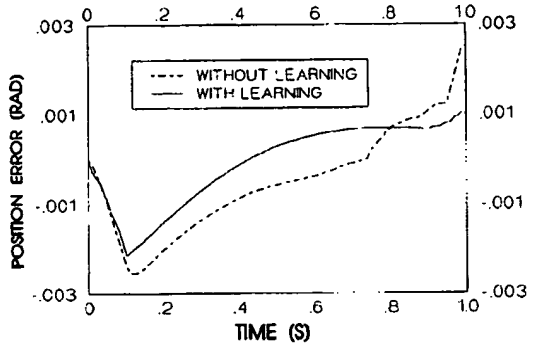

Fig. 14. Position error for the third d.o.f. in generalization phase.

The generalization results (position errors with learning using RLS method and without learning for the first and third joint) are presented in Figs. 13 and 14.

We can see that with "learned" inverse dynamic robot model, tracking for quite different robot trajectories is quite satisfactory.

VI. Conclusion

The results presented indicate the feasibility of using highly efficient connectionist approaches for learning complex input-output relations of robot's dynamics. However, naive straightforward neural network is not suitable for practical solutions in real-time, because of the increased dimensionality of input and output spaces and the long learning time. Hence, based on this facts proposed new connectionist control algorithms with decomposed "*3F-2SF*" AND "*3F-1SF*" structure use available information about robot dynamic but only in general and specific form. The use of decomposition principle in the space of internal robot coordinates enables the synchronous training of several multilayer perceptrons on the simpler input-output relations with significant reduction of learning time.

We have studied acceleration of the learning algorithms for multilayer perceptrons in robot control and proposed a new algorithms that uses Recursive Least Square method and Kalman Filter approach for estimating of weighting factors in network layers. These algorithms have the feature that the learning rate is time dependent, which enables that with sufficiently convergent solution they assures faster learning than the generalized delta rule in standard backpropagation algorithms. Also, as result, adaptive capability of the connectionist controller to the system uncertainties was clarified.

The proposed algorithms for connectionist robot learning are strictly related for the neural networks as static nonlinear input-output maps. For future research, it is very attractive idea of the using reccurent neural networks as well as process of evolution in the static multi-layer perceptrons. The neural networks that have recurrent nature, have potentially more computational power than pure feedforward neural networks. This type of neural networks are very promising for the future exploitation in robotic system, because these networks can solve some additional problems of system stability. At the other side, change in the neural network structure (propagation of neurons and degeneration of synapses) during the training process enable dynamic adaptation of the network capacity to the complex problem of the network interaction with the environment.

The proposed neuromorphic structure with decomposition principle and fast convergence properties has a great possibility in real-time control and learning of manipulation robots.

Acknowledgment

Special appreciation is expressed to Prof. Srdjan Stanković for many helpful discussions in the field of neural networks.

The authors also gratefully acknowledges several helpful comments and suggestions of the anonymous reviewers.

References

[1] G. N. Saridis, "Intelligent robotic control," *IEEE Transactions on Automatic Control* vol. AC-28. no. 5, pp. 547–557, 1983.

[2] A. Meystel "Intelligent control in robotics," *Journal of Robotic Systems*, vol. 5, no. 4, pp. 269–308, 1988.

[3] D. Katić and M. Vukobratović, "Decomposed connectionist architecture for fast and robust learning of robot dynamics," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Nice), pp. 2064–2069, May, 1992.

[4] S. Y. Kung and J.-N. Hwang, "Neural network architectures for robotic applications," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 5, pp. 641–657, 1989.

[5] D. Psaltis, A. Sideris and A. Yamamura, "A hierarchical model for voluntary movement and its application to robotics," in *Proceedings of the 1st IEEE International Conference on Neural Networks* San Diego, pp. 551–558, June, 1987.

[6] W. T. Miller, R. P. Hewes, F. H. Glanz and L. G. Kraft, "Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 1, pp. 1–9, 1990.

[7] A. Guez, J. Eilbert and M. Kam, "Neuromorphic architectures for fast adaptive robot control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Raleigh), pp. 145–149, April 1988.

[8] D. F. Bassi, "Connectionist Dynamic Control of Robot Manipulators," *Ph.D.* University of Southern California, Los Angeles, December 1990.

[9] D. Y. Yeung and G. A. Bekey, "Using a context-sensitive learning network for robot arm control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Scottsdale), pp. 1441–1447, May, 1989.

[10] M. Kawato, K. Furukawa and R. Suzuki, "A hierarchical neural network model for control and learning of voluntary movement," *Bilogical Cybernetics*, vol. 57, pp. 169–185, 1987.

[11] H. Miyamoto, M. Kawato, T. Setoyama and R. Suzuki, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," *Neural Networks*, vol. 1, pp. 251–265, 1988.
[12] M. Kawato, Y. Uno, M. Isobe and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to roboticsi," *IEEE Control Systems Magazine*, vol. 8., no. 2, pp. 8–16, 1988.
[13] D. Katić, "Connectionist learning models for Intelligent control of manipulation robots," in *Proceedings of the IMACS International Symposium on Mathematical and Intelligent Models in System Simulation*, (Brussels), pp. I.C.3-1–I.C.3-6, September 1990.
[14] K. Goldberg and B. Pearlmutter, "Using a neural network to learn the dynamics of the CMU direct-drive arm II," *Tech. Rep. CMU-CS-88-160*, Carnegie Mellon University, Pittsburgh, August 1988.
[15] T. Ozaki, T. Suzuki, T. Furuhashi, S. Okuma and Y. Uchikawa, "Trajectory control of robotic manipulators," *IEEE Transations on Industrial Electronics*, vol. 38, no. 3, pp. 195–202, 1991.
[16] J. K. Kruschke and J. R. Movellan, "Benefits of gain: speeded learning and minimal hidden layers in back-propagation networks," *IEEE Transations on Systems, Man and Cybernetics*, vol. 21, no. 1, pp. 273–280, 1991.
[17] M. Vukobratović, D. Stokić and N. Kirćanski, "Non-adaptive and adaptive control of manipulation robots," Springer Verlag, Berlin, 1985.
[18] S. Arimoto, S. Kawamura and F. Miyazuki, "Bettering operation of robots by learning," *Journal of Robotic Systems*, vol. 1, pp. 123–140, 1984.
[19] N. Sadegh, R. Horowitz, W.-W. Kao and M. Tomizuka, "A Unified approach to design of adaptive and repetitive controllers for robotic manipulators," in *Proceedings of the USA-Japan Symposium on Flexible Automation*, (Minneapolis), pp. 223–231, 1988.
[20] C. G. Atkeson and D. J. Reinkensmeyer, "Using associative-addressable memories to control robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Scottsdale), pp. 1859–1864, May, 1989.
[21] H. Asada, "Teaching and learning of compliance using neural nets: representation and generation of neural compliance," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Cincinnati, pp. 1237–1244, May, 1990.
[22] H. Liu, T. Iberall and G. A. Bekey, "Neural network architectures for robot hand control," *IEEE Control Systems Magazine*, vol. 9, no. 2, pp. 38–43, 1989.
[23] D. E. Rumelhart and J. L. McClelland, "Parallel distributed processing (PDP): Exploration in the microstructure of cognition," vol. 1–2, Cambridge: MIT Press, 1986.
[24] D. F. Bassi and G. A. Bekey, "Decomposition of neural networks models of robot dynamics: a feasibility study," in *Simulation and A.I.*, (W. Webster, ed.), pp. 8–13, The Society for Computer Simulation International, 1989.
[25] M. J. Quinn, "Designing efficient algorithms for parallel computers," New York, McGraw-Hill, New York, 1987.
[26] C. J. Wang and C.-H. Wu, "Parallel simulation of neural networks," *Simulation*, vol. 56, no. 4, pp. 223–232, 1991.
[27] R. Hecht-Nielsen, "Neurocomputing," Addison-Wesley, Reading: 1990.
[28] G. Agha, "Actors, A Model for concurrent Computation in Distributed Systems" Cambridge, The MIT-Press, 1986.
[29] T. Tollenaere, "SuperSab:Fast adaptive back propagation with good scaling properties," *Neural Networks*, vol. 3, no. 5, pp. 561–573, 1990.
[30] R. A. Jacobs, "Increased rates of convergence through learning speed in back-propagation networks," *Neural Networks*, vol. 1, no. 2, pp. 295–307, 1988.
[31] M. K. Weir, "A method for self-determination of adaptive learning rates in back propagation," *Neural Networks*, vol. 4, no. 3, pp. 371–379, 1991.
[32] A. K. Rigler, J. M. Irvine and T. P. Vogl, "Rescaling of variables in back propagation learning," *Neural Networks*, vol. 4, no. 2, pp. 225–229, 1991.
[33] L. Ljung and T. Soderstrom, "Theory and Practice of Recursive Identification." Cambridge: The MIT-Press, 1983.
[34] S. Stefunović and B. Kovačević, "Neural networks learning using recursive least-squares method", *Proceedings of the IASTED Conference on Signal Processing and Digital Filtering*, Lugano, 1990, pp. 551–556.
[35] K. Watanabe, T. Fukuda and S. G. Tzafestas, "Learning algorithms of layered neural networks via extended kalman filters," *International Journal of Systems Science*, vol. 22, no. 4, pp. 753–768, 1991.
[36] Y. Iiguni, H. Sakai and H. Tokumaru, "A real-time learning algorithm for a multilayered neural network based on the extended kalman filter," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 959–966, 1992.
[37] S. Turk and M. Otter, "The DFVLR Models 1 and 2 of the Manutec r3 robot," *Robotersysteme*, no. 3, pp. 753–768, 1987 (in German).

Duško Katić received the B.S. and M.S. degrees in mechanical engineering, and Ph.D. degree in electrical engineering from the University of Belgrade, Yugoslavia in 1982, 1987, and 1994, respectively.

Since 1983, he has been a researcher with the Robotics Department, Mihajlo Pupin Institute, Belgrade, where he is involved in the modeling and design of advanced control systems for manipulation robots. He is the author of about 20 scientific papers from the field of robotics. His current scientific interests include application of intelligent control for large scale dynamic systems with specific emphasis on adaptive and learning control algorithms.

Miomir Vukobratović was born in Zrenjanin, Yugoslavia in 1931. He received the B.Sc. and Ph.D. degrees in mechanical engineering form the University of Belgrade in 1957 and 1964, respectively, and the D.Sc. degree from the Institute Mashinivedenya, Russian Academy of Sciences, Moscow, 1972. He is presently director of the Robotics Centre at Mihajlo Pupin Institute, Belgrade. He is visiting professor teaching graduate courses in robotics at several universities in Yugoslavia and abroad.

His main interest is in the development of efficient modeling of robotic systems dynamics. His special interest being dynamic non-adaptive and adaptive control of non-contact and contact tasks in robotics. He is author or co-author of 160 scientific papers in the field of robotics published in leading international journals. He is author or co-author of 16 monographs published in English, Japanese, Russian and Serbian.

He is chairman of the Robotics Section, Yugoslav Society for Electronics, Telecommunications, Automation and Nuclear Engineering. He is scientific leader of the Serbian national program in robotics, as well as the principal investigator of three international robotics projects (EC, NSF, UNIDO). M. Vukobratović is a permanent member of international committees of several IFAC, IFAC/IFIP and IFToMM symposia. He is also for many years a member of editorial board of several leading scientific journals in robotics, manufacturing and artificial intelligence. He is a member of ASME and a senior member of IEEE. He is also a member of the Scientific Society of Serbias, a member of the Serbian Academy of Sciences and Arts since 1981, and foreign member of Russian Academy of Sciences since 1988, and member of the International Academy of Engineering since 1994. He was the winner of the highest Yugoslav state award in 1982 for his outstanding achievements in robotics and technical cybernetics. He is also, with his co-authors, the winner of the highest scientific Yugoslav award, "Nikola Tesla", in 1986.